UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/886,178	06/20/2001	David Wallman	SUN1P830/P6124	5987

22434 7590 06/06/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 06/06/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/886,178	WALLMAN ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *20 June 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *02/22/05*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to amendment dated February 05, 2004.

2. Per Applicants' request, the Specification, claims 1, 6-8, 11-20, 22-23 have been amended, and Claim 2 is canceled.

3. Claims 1-25 remain pending.

Response to Amendment

4. Applicants' amendment dated 02/18/2005, responding to the 11/30/2004 Office action provided in the objection of specification, CROSS-REFERENCE TO RELATED APPLICATIONS. The examiner has reviewed the updated specification respectfully.

5. The objection to the Specification is hereby withdrawn in view of Applicants' amendment to the specification.

6. Applicants' amendment dated 02/18/2005, responding to the 11/30/2004 Office action provided in the objection of specification, the use of the trademark JAVA. The examiner has reviewed the updated specification respectfully.

7. The objection to the Specification is hereby withdrawn in view of Applicants' amendment to the Specification, the use of the trademark JAVA. However, there still are 'Java' referred in the application, such as in claims 1, 4, and 20, see Claim Objections below.

8. Applicants' amendment dated 02/18/2005, responding to the 11/30/2004 Office action provided in the 35 USC § 101 rejection . The examiner has reviewed the updated specification respectfully.

9. The 35 USC § 101 rejections to Claims 11-19 is hereby withdrawn in view of Applicants' amendment to the specification.

10. Applicants' amendment dated 02/18/2005, responding to the 11/30/2004 Office action provided in the 35 USC § 112 rejection . The examiner has reviewed the updated specification respectfully.

11. The 35 USC § 112 rejections to Claims 1,2,5, 11 and 20 is hereby.withdrawn in view of Applicants' amendment to the specification.

Response to Arguments

12. Applicants' arguments for Claims 1-18, 20-25 have been fully considered respectfully by the examiner but they are not persuasive.

13. Applicants' arguments are basically in the following points:

- "optimizing the native code is not the same as optimizing the runtime environment", Cyran does not teach optimizing the runtime environment. (Remarks, page 8, 4th paragraph).

  Examiner's Response: In response to applicant's argument that Cyran's disclosure teaches optimizing the native code, does not teach optimizing the runtime environment. The Applicant's argument is not correct. Cyran's disclosure also teaches to optimize a runtime environment. See Cyan's column 2, lines 20-22, "In yet another aspect of the invention, the **optimization information** is provided to the code interpretive **runtime system** an **attributes added to class files**", and column 4, lines 48-50, "The presence of this **attribute** in the class file 14 informs the **Java runtime system** to JIT compile the intermediate codes for the method that includes the 'Code' **attribute**." - it's adding class attribute information to the runtime environment, which optimizes the runtime performance. Cyan's disclosure

does not just optimizing the native code. Also see FIG. 1, the item 12, is the code preparation system (where optimizing the native code), combining item 14, the 'Extended Class File', where particular features (attributes) for a particular application is added to the runtime environment. Therefore, Cyan's disclosure teaches both native code optimization and the runtime environment optimization.

- With regard to claims 11 and 20, given the relevant similarities between claims 11 and 20, and claim 1, same response is given as above.

14. Examiner is maintaining the 35 USC § 103 Rejections. For the Applicants' convenience they are listed as following, with the amendments requested by the Applicants.

Claim Objections

15. Claims 1, 4, and 20 are objected to because of the following informalities: 'Java' is used instead of 'JAVA'. Appropriate correction is required.

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. Claims 1, 3-18, 20-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Robert J. Cyran et al. U.S. Patent No. 6,412,107 (hereinafter "Cyran"), in view of Maynard Paul Johnson et al. U.S. Patent No. 6,330,709 (hereinafter "Johnson").

CLAIM	Cyran / Johnson
1. In a JAVA computing environment, a method of generating optional attributes in a JAVA class file, said method comprising: (a) receiving as input a JAVA runtime environment optimization information; (b) generating one or more optional attributes based on said Java runtime environment optimization information; and (c) writing said one or more optional attributes in an attribute table portion of a Java class file.	For items (a) and (b), in Cyran, refer to FIG.4, and see column 6, lines 16-20, "operation continues at decision block 56 where the user may choose to **perform optimization analysis** on the **input code** *(receiving input Java code)*. If **selected** (*optional*), operation continues to block 58 where the code analyzer 36 is invoked to pre-process, i.e., pre-compile the **input code**." Further, in Cyran column 4, lines 48-55, "the presence of this **attribute in the class file** 14 informs the **Java runtime system** to JIT compile the intermediate codes for the method that includes the 'Code' **attribute**. In addition, the COM.TexasInstruments.JIT **attribute** is also used to pass **optimization information** generated by the code preparation system 12 or directions to the JIT compiler instructing it to perform its own **optimizations at compile time**, as discussed hereinabove." For item (c), in Cyran column 2, lines 51-54, "The **input code** 11 may also be source code, such as **Java source code**, which is first translated into intermediate code format before being analyzed. The **optimization information** it provided as additional **attributes** added to **class files** 14

generated by the code preparation system 12." Also Cyran column 8, lines 49-55, "Other optimization information passed includes a fully annotated flowgraph or common sub-expression information useful in cases where a certain object's fields or methods are repeatedly accessed or invoked. Note also that the given optimization information can be either target dependent or target independent and used to create one or more sets of optimization attributes (*attribute generator*). Cyran teaches all aspects of claim 1, but he does not mention 'attribute table' specifically, however, Johnson teaches it in an analogous prior art. In Johnson, column 5, lines 28-32, "Another approach which provides object persistence is 'externalization'. Externalization is the means or protocol used in object-oriented programming for transferring data out of an object. In essence **the 'state data' that defines the attributes of an object** are "externalized", or written out of the object, **into a different format that is easily stored in the local data store.** (*attribute table*)" Also, in column 11, lines 53-67, "When the virtual address translator 210 receives an SAS address to translate it runes the SAS address through hasher 215 to generate a key number n. The hash **table** 216, preferably a list of pointers to page **table** entry lists, is then searched to locate the page **table**

entry list in the lookaside buffer 218 corresponding to that key number n. The page **table** entry list corresponding to that key number n in the lookaside buffer 218 is then searched to determine if requested data is in the page cache 212." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Cyran's disclosure of the optimization Java environment by writing the attributes to an attribute table taught by Johnson, for the purpose of retrieving for data object from the storage (Johnson, column 12, lines 6-7).

2. (Cancelled)

3. A method as recited claim 1, wherein said method further comprises:
generating computer program code that implements an application programming interface suitable for loading said one or more optional attributes.

For the feature of claim 1 see claim 1 rejection. In Cyran, column 6, lines 3-5, "The code preparation system (for *computer program code*) of the present invention provides developers with an **interface** that allows them to **select** (*loading*) these individual optimizations."

4. A method as recited claim 3, wherein said application programming interface can be used to read said one or more optional attributes from said Java class file.

For the feature of claim 3 see claim 3 rejection. See claim 2 rejection (*retrieving data object*), further in Johnson, column 4, lines 56-58, "the only way to **retrieve** *(read),* process or otherwise operate on the object is through the methods defined on the object."

5. A method as recited claim 4, wherein said application programming interface includes functions that can be used to read first, last, and next optional attributes in said JAVA class file.	For the feature of claim 4 see claim 4 rejection. See claim 2 rejection, where the prior art visit every object in the table; which includes **first**, **last** and **next** object in the table.
6. A method as recited claim 4, wherein said application programming interface includes a function suitable for finding an optional attribute in said JAVA class file.	For the feature of claim 4 see claim 4 rejection. See claim 3 rejection, where the prior art teaches to 'select' an object, which implies 'finding' it.
7. A method as recited claim 1, wherein said JAVA runtime optimization is stored in a database.	For the feature of claim 1 see claim 1 rejection. See Cyran FIG. 2, where 'storage device' can be a database, also in claim 1 (c) rejection, the attributes are stored in a 'table' which can also be a database (see Johnson FIG. 2).
8. A method as recited in claim 7, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.	For the feature of claim 7 see claim 7 rejection. See Cyran FIG. 1, the Extended Class File is an extension of the Code Preparation System, which is suitable for analyzing a Java application. Also see Johnson FIG. 2, the Storage System is an extension of the Java Virtual Machine (*compiler*), which is also suitable for analyzing a Java application.
9. A method as recited in claim 7, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.	For the feature of claim 7 see claim 7 rejection. See Johnson column 2, lines 33-38, "When a process needs to access a persistent object, the process must contact the **file manager** which locates the persistent object data in a file on

	backing store and move a copy of the persistent object data into a memory buffer. The persistent object data must then be **reconstructed** (*generate and load*) into a persistent object in memory."
10. A method as recited in claim 7, wherein said method further comprises: updating said database to reflect generation of said one or more optional attributes.	For the feature of claim 7 see claim 7 rejection. See Johnson, column 2, lines 22-23, "The file manager **stores** (*updating*) **and retrieves data** from the permanent storage devices in the form of files."
11. In a JAVA computing environment, a JAVA optional attribute generator computer-implemented method suitable for generation of optional attributes in a JAVA class file, said JAVA optional attribute generator computer-implemented method operating to: receive as input a JAVA runtime environment optimization information; generate one or more optional attributes based on said JAVA runtime environment optimization information; and write said one or more optional attributes in an attribute table portion of a JAVA class file.	Same as claim 1 rejection; the 'code preparation system' disclosed in Cyran's prior art functions as an 'attribute generator'.
12. A JAVA optional attribute generator as recited in claim 11, wherein said JAVA optional attribute generator computer-implemented method operates to generate	For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 3 rejection.

computer program code that implements an application programming interface suitable for loading said one or more optional attributes.

13. A JAVA optional attribute generator as recited in claim 11, wherein an application programming interface can be used to read said one or more optional attributes from said JAVA class file.

For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 4 rejection.

14. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said JAVA runtime environment optimization information is stored in a database.

For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 7 rejection.

15. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.

For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 8 rejection.

16. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.

For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 9 rejection.

17. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said optional attribute generator operates to update said database to reflect generation of said one or more optional attributes.	For the feature of claim 11 see claim 11 rejection. For the rest of the feature see claim 10 rejection.
18. A JAVA optional attribute generator computer-implemented method as recited in claim 11, wherein said optional attribute generator operates to generate a description of an optional attribute.	For the feature of claim 11 see claim 11 rejection. In Cyran, column 8, lines 57-64, "The code preparation system 12 also passes **directions** (*description*) to the code interpretive runtime system in addition to passing optimization information. These **directions** are usually relating to optimizations that must be done on generated native code as opposed to being done on intermediate code. Thus, for example, **directions** can be given to the JIT compiler to perform **instruction** scheduling and peephole optimizations as described hereinabove in the **description** of the COM.TexasInstruments.JITOptimizations attribute."
20. A computer readable medium including computer program code for generating optional attributes in a JAVA class file, said computer readable medium comprising: (a) computer program code for receiving as input a JAVA runtime environment optimization information; (b) computer program code for generating one or more optional	Same as claim 1 rejection; a readable medium is disclosed in Cyran's prior art, see FIG. 2.

attributes based on said Java runtime environment optimization information; and

(c) computer program code for writing said one or more optional attributes in an attribute table portion of a Java class file.

21. A computer readable medium as recited in claim 20, wherein said method further comprises:

generating computer program code that implements an application programming interface suitable for loading said one or more optional attributes.

For the feature of claim 20 see claim 20 rejection. For rest of the features see claim 3 rejection.

22. A computer readable medium as recited in claim 21, wherein said JAVA runtime environment optimization inforamtion is stored in a database.

For the feature of claim 21 see claim 21 rejection. For rest of the features see claim 7 rejection.

23. A computer readable medium as recited in claim 22, wherein said database is generated by a compiler extension or a software tool suitable for analyzing a JAVA application.

For the feature of claim 22 see claim 22 rejection. For rest of the features see claim 8 rejection.

24. A computer readable medium as recited in claim 22, wherein said database is stored in a runtime performance manager that can interact with software modules that generate and load said one or more optional attributes.

For the feature of claim 22 see claim 22 rejection. For rest of the features see claim 9 rejection.

25. A computer readable medium as recited in claim 24, wherein said method further comprises: updating said database to reflect generation of said one or more optional attributes.	For the feature of claim 24 see claim 24 rejection. For rest of the features see claim 10 rejection.

18. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Robert J. Cyran et al. U.S. Patent No. 6,412,107 (hereinafter "Cyran"), in view of Maynard Paul Johnson et al. U.S. Patent No. 6,330,709 (hereinafter "Johnson"), and further in view of Cary Lee Bates et al. U.S. Patent No. 6,769,015 (hereinafter "Bates").

CLAIM	**Cyran / Johnson / Bates**
19. A JAVA optional attribute generator computer-implemented method as recited in claim 18, wherein said description is in XML format.	For the feature of claim 18 see claim 18 rejection. Cyran and Johnson teach all aspects of claim 19, but he does not mention 'in XML' specifically, however, Bates teaches it in an analogous prior art. In Bates' column 6, lines 22-23, "The attributes may be encoded using, e.g., HTML or XML (extensible markup language)". It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Cyran and Johnson's disclosure of the Attributes Generator in a Java environment by using XML taught by Bates, for the purpose of retrieving the attribute information (Bates column 6, lines 21).

Conclusion

19. The following summarizes the status of the claims:

35 USC § 103 claim rejection: 1-25

20. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow

Examiner

Art Unit 2192

May 31, 2005

CC

ANTONY NGUYEN-BA
PRIMARY EXAMINER